Memfault

The Power of Metrics

Tyler Hoffman

Tyler Hoffman

Co-Founder & Head of Developer Experience, Memfault

- Passions: tooling and automation in firmware engineering
- Previously Firmware Engineers @ Pebble and Fitbit
- Can find their thoughts and content on Memfault's Interrupt blog (interrupt.memfault.com)

pebble Memfault fitbit

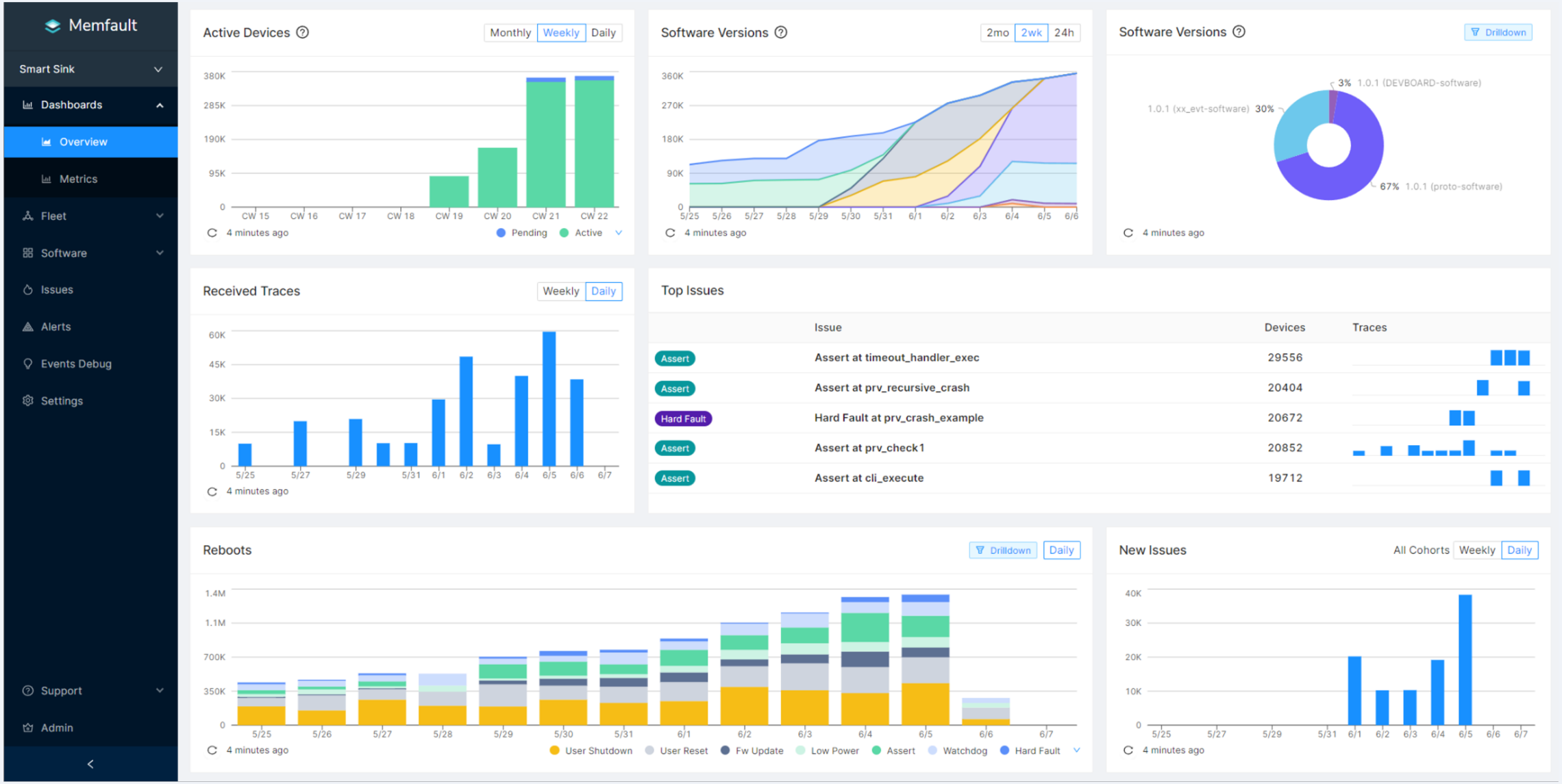
Agenda

1. What & Why of Metrics
2. Battery Life Metrics
3. Power Consumption Metrics
4. Connectivity Metrics
5. Best Practices

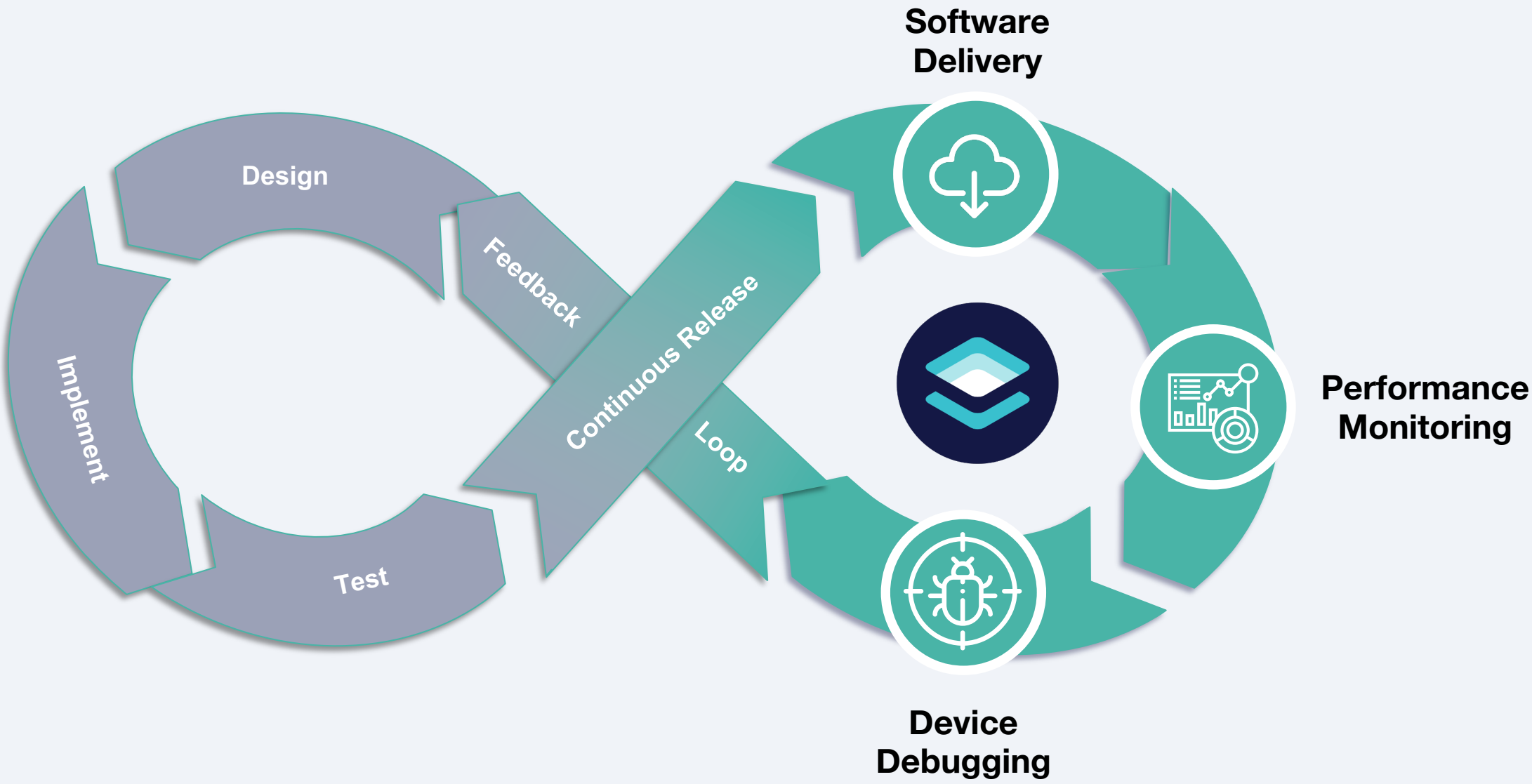
Memfault's Relationship to Metrics

One of the three pillars of the Memfault offering

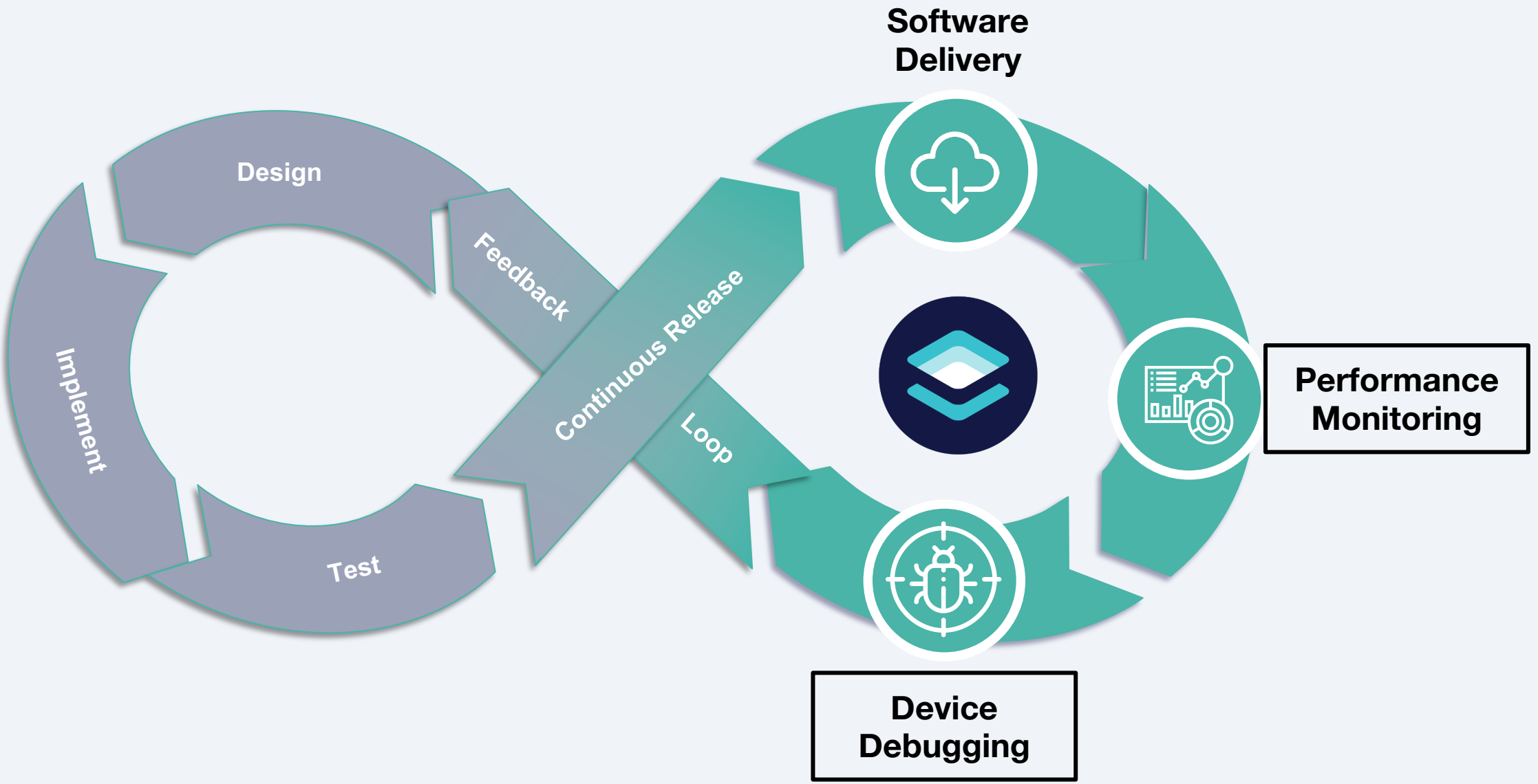
Memfault for Device Reliability Engineering



Memfault for Device Reliability Engineering



Memfault for Device Reliability Engineering



Agenda

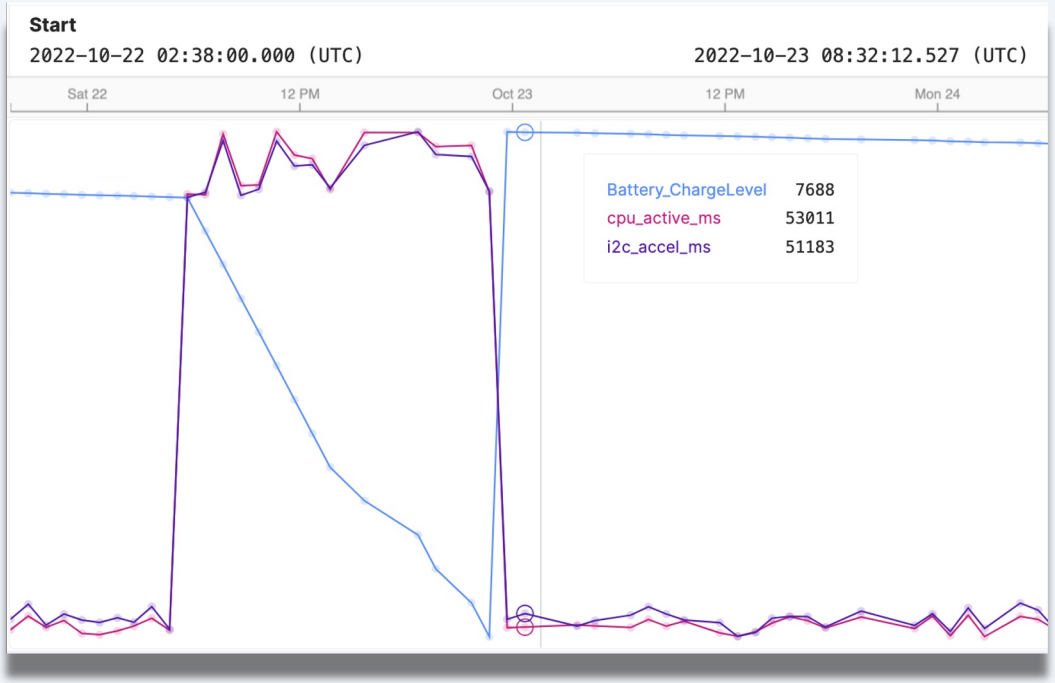
1. What & Why of Metrics
2. Battery Life Metrics
3. Power Consumption Metrics
4. Connectivity Metrics
5. Best Practices

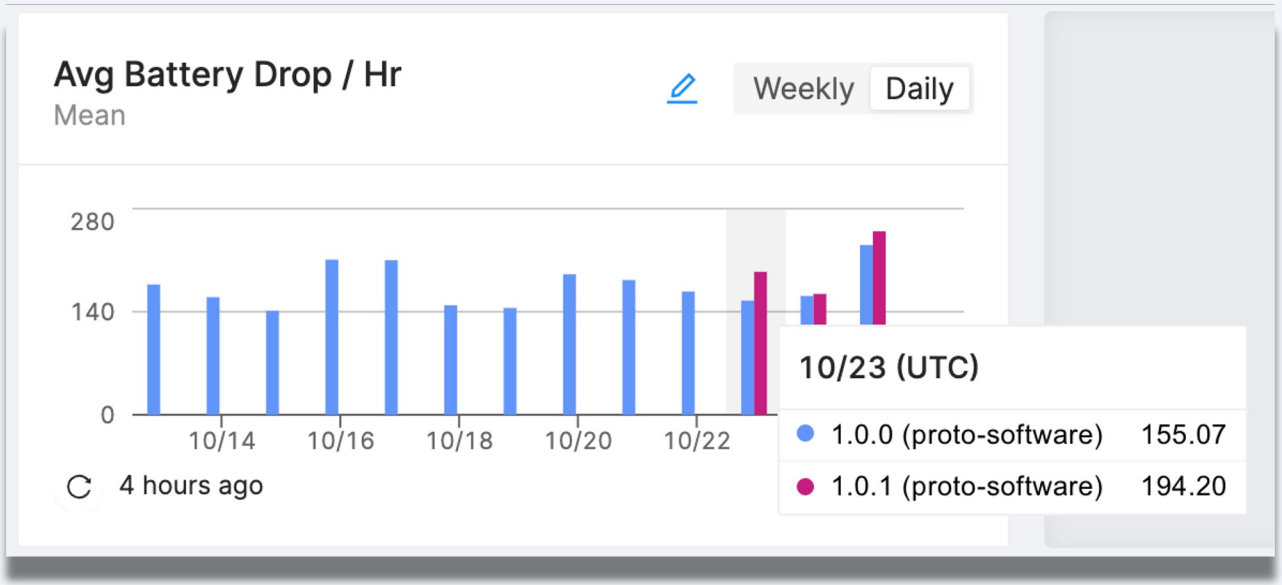
What & Why of Metrics

Let’s cover the basics first

Metrics

A **metric** is a measurement captured at runtime





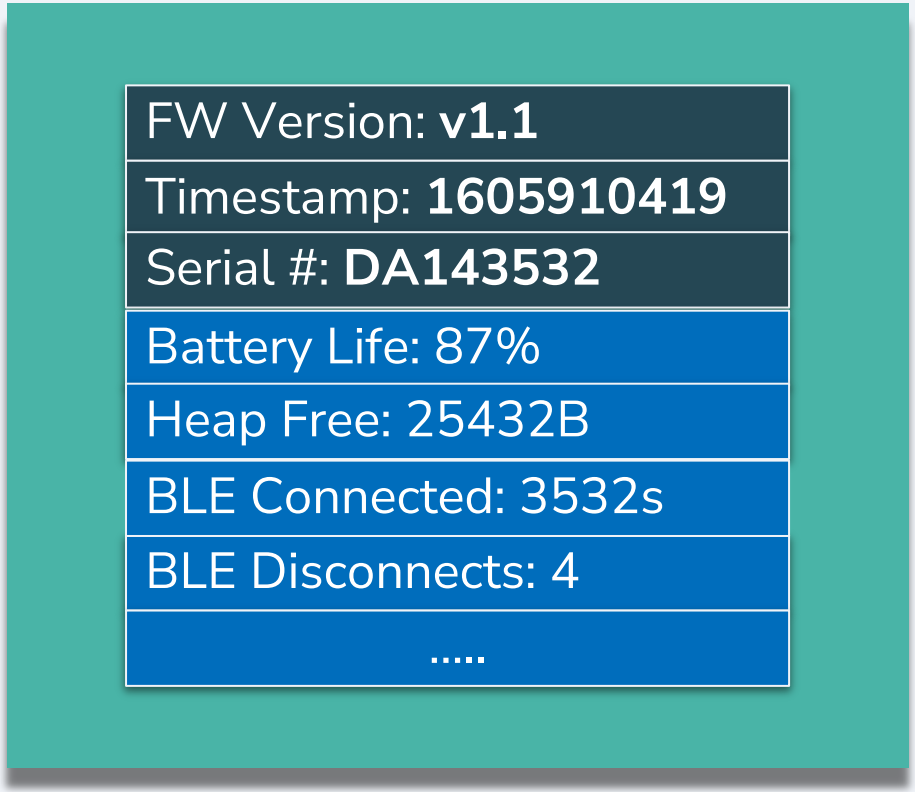
Combing large numbers of metrics and calculating statistics is called an **aggregation**

Use of Metrics

Uncover trends on anything that is numerical

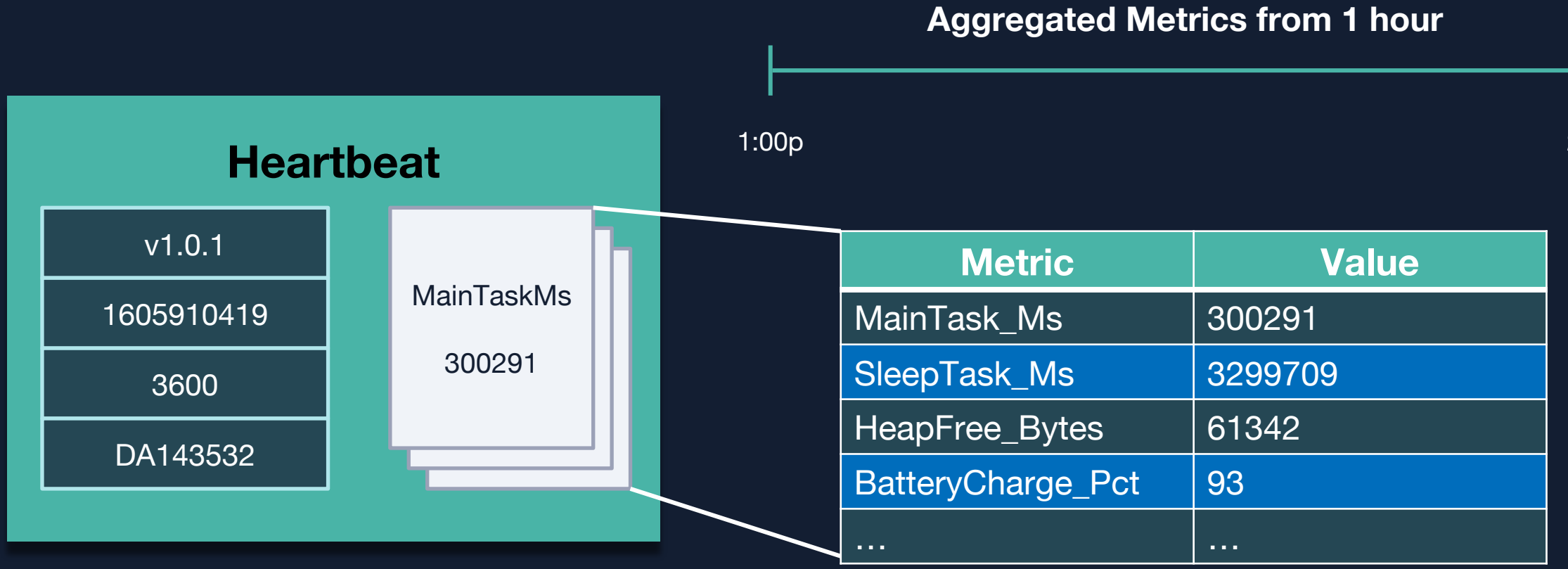
Common metrics for firmware developers are:

- Task runtimes
- Heap information
- Connectivity statistics
- Frequency of errors
- Peripheral utilization and power states
- Many, many more.



Pinpoint power, connectivity, and performance issues and regressions

A Metric Heartbeat



https://interrupt.memfault.com/blog/device-heartbeat-metrics

A Metric Heartbeat

Metric ID Definitions

```
// Header
MEMFAULT_METRICS_KEY_DEFINE(FlashBytesWritten,
                            kMemfaultMetricType_Unsigned);

MEMFAULT_METRICS_KEY_DEFINE_WITH_RANGE(BatteryPercent,
                                       kMemfaultMetricType_Unsigned
                                       0, 100);
```

Metric ID’s - unsigned integers (**1-4 bytes)**

Metric values - integers or floats **(4 bytes)**

Metrics API

```
// Header
MEMFAULT_METRICS_KEY_DEFINE(FlashBytesWritten,
                            kMemfaultMetricType_Unsigned);
```

```
// Counters
void memfault_metrics_heartbeat_add(MemfaultMetricId key,
                                    int32_t value);

// Timers
void memfault_metrics_heartbeat_timer_start(MemfaultMetricId key);
void memfault_metrics_heartbeat_timer_stop(MemfaultMetricId key);

// Gauges
void memfault_metrics_heartbeat_set_signed(MemfaultMetricId key,
                                           int32_t value);
void memfault_metrics_heartbeat_set_unsigned(MemfaultMetricId key,
                                             uint32_t value);
```

Simple API to track integers

Metric Counters

```
void flash_sector_erase(uint16_t sector) {
    ...
    memfault_metrics_heartbeat_add(MEMFAULT_METRICS_KEY_DEFINE(FlashSectorErases), 1);
}

void flash_write_bytes(uint32_t addr, void *buf, size_t n) {
    ...
    memfault_metrics_heartbeat_add(MEMFAULT_METRICS_KEY_DEFINE(FlashWriteBytes), n);
}
```

Count anything with 2 lines of code using the Memfault SDK

Metric Timers

```c
static uint32_t s_wifi_ticks;

void wifi_connected_callback(void) {
    // Start timer when Wi-Fi connects
    memfault_metrics_heart_timer_start(MEMFAULT_METRICS_KEY_DEFINE(WifiConnectedDuration));
}

void wifi_disconnected_callback(void) {
    // Stop timer when Wi-Fi connects
    memfault_metrics_heart_timer_stop(MEMFAULT_METRICS_KEY_DEFINE(WifiConnectedDuration));
}
```

Enables tracking the time of states or operations

Metric Gauges

```
void memfault_metrics_heartbeat_collect_data(void) {
    ...
    static int32_t s_prev_battery_pct;

    const int32_t current_battery_pct = battery_get_pct();
    const int32_t battery_delta =
        current_battery_pct - s_prev_battery_pct;

    memfault_metrics_heartbeat_set_unsigned(MEMFAULT_METRICS_KEY_DEFINE(BatteryPctDrop),
                                            battery_delta);

    memfault_metrics_heartbeat_set_unsigned(MEMFAULT_METRICS_KEY_DEFINE(BatteryPct),
                                            current_battery_pct);
}
```

Sends current values or

metrics tracked outside of counters or timers

Poll #1

How many metrics does your team collect from remote production devices?

A. None

B. 1-10

C. 11-50

D. 51+

Agenda

1. What & Why of Metrics
2. Battery Life Metrics
3. Power Consumption Metrics
4. Connectivity Metrics
5. Best Practices

Battery Life Metrics

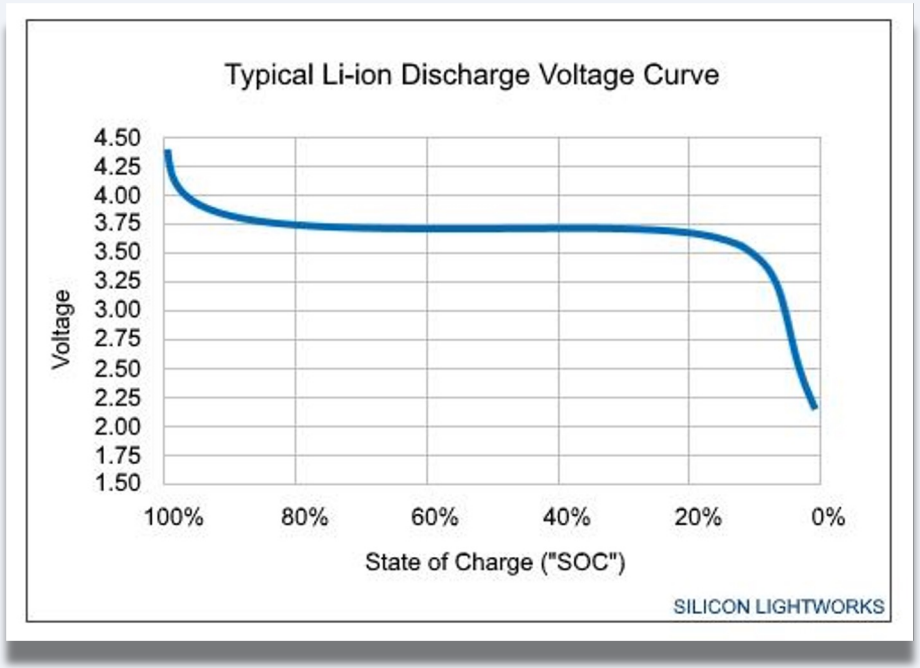
Wrangle battery life projectections

Battery Life in Hardware Products

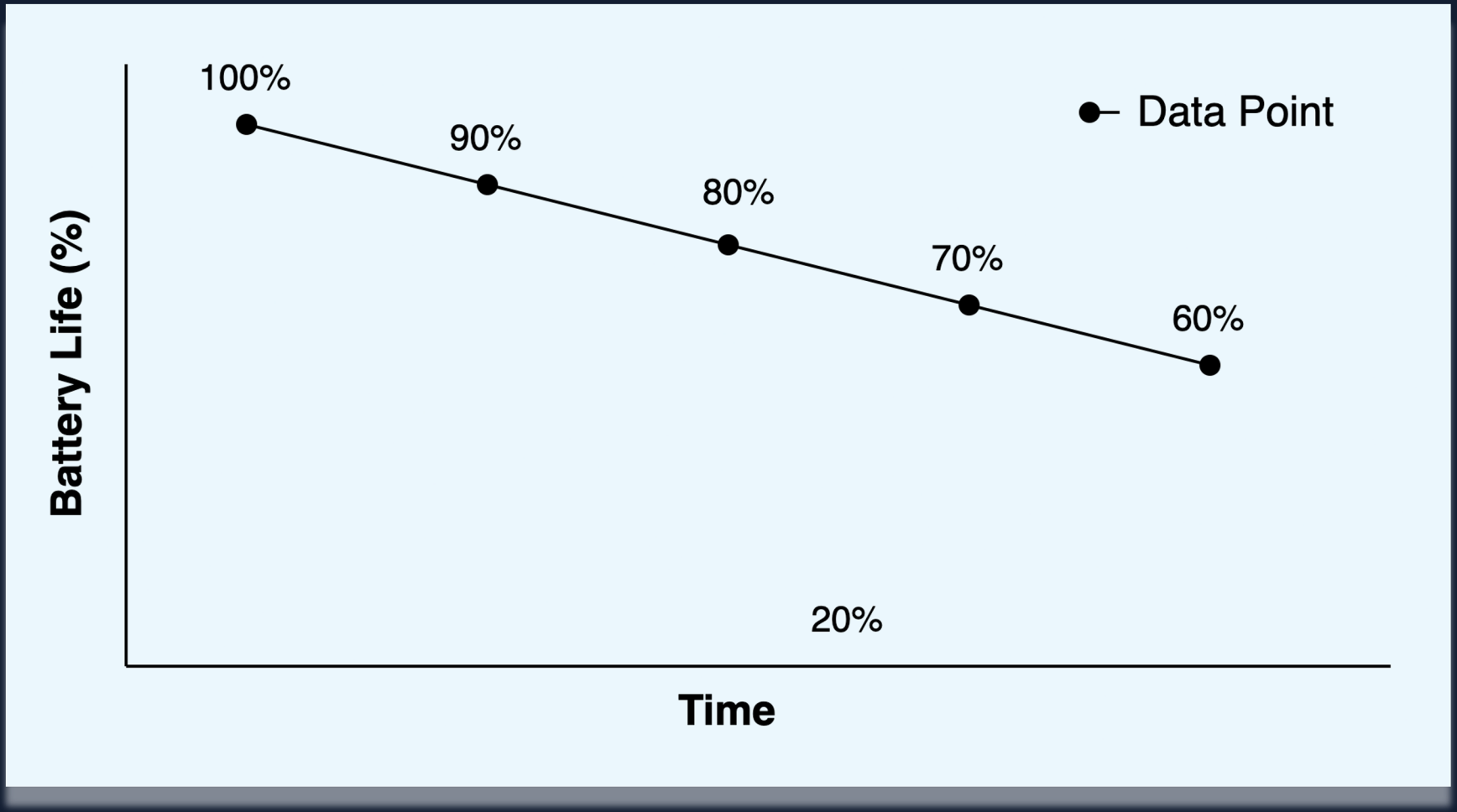
- **Critically important to customers**
- If it's out of power, it's non-functional
- Pebble Watch
  - 7 days on the box,
    - ~130mAh battery
  - 2 days on the box,
    - ~56 mAh battery
  - Primary differentiator from competitors
  - #1 or #2 support issue from customers
  - We spent **a lot of engineering time** on battery life

Reasons why batteries might lie to you

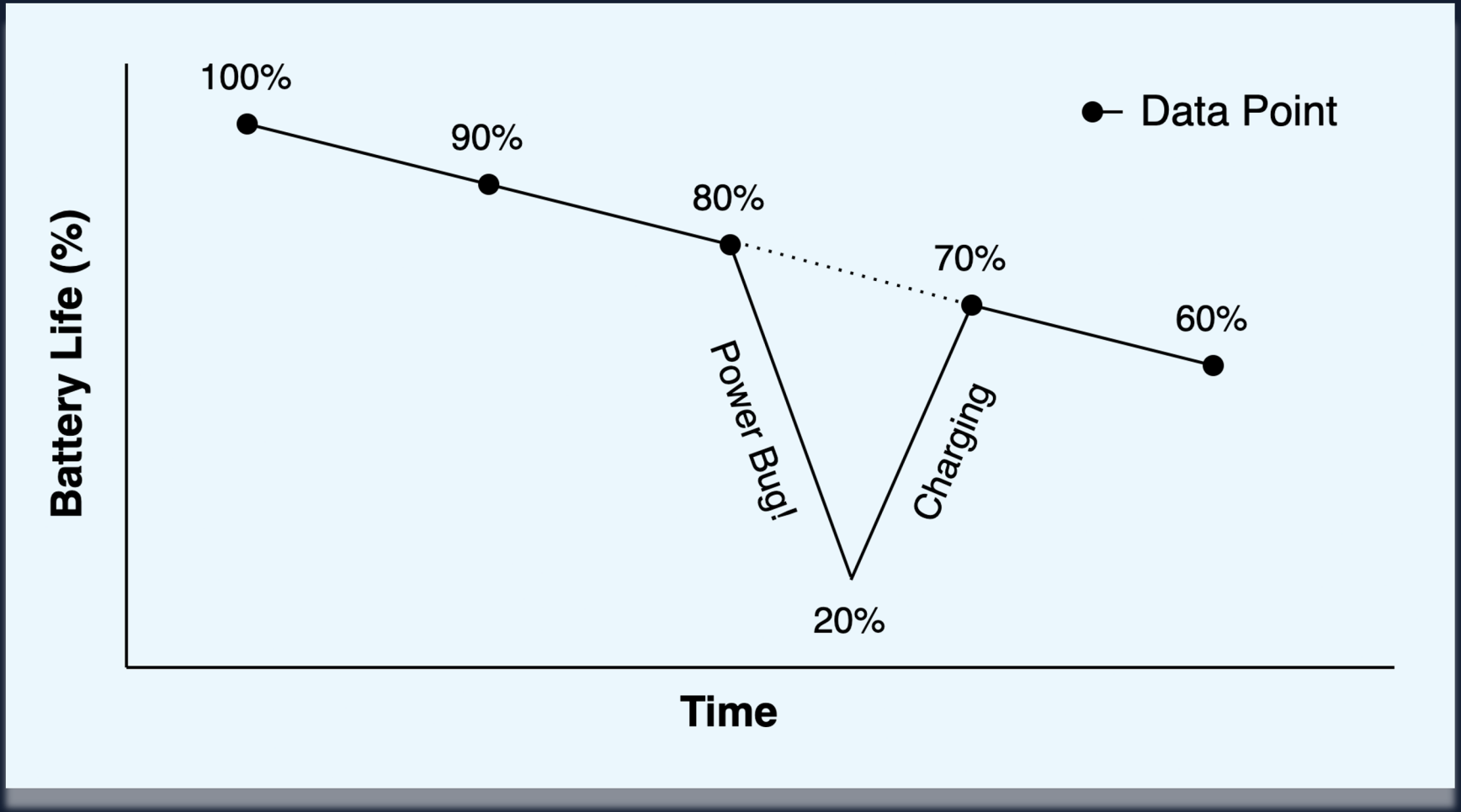
- Battery % requires a good battery curve
  - Takes many batteries and iterations to get one
- Querying during a high power draw might give false value
  - High power draw results in lower mV readings
  - 90% → 30% → 80%
- Temperature affects battery readings
- Batteries curves differ between batches
- Batteries age over time



Battery Life Metrics are tricky



Battery Life Metrics are tricky



Tracking only current battery % hides information

Calculating Average Battery Life of Devices

Device A	Device B	Device C	Device D
75%	23%	92%	5%
72%	21%	89%	2%
67%	19%	85%	**10%**
34%	19%	**100%**	7%
78%	**21%**	97%	5%
50%	**100%**	94%	**15%**
100%	92%	92%	12%

bold = % increase

Compute battery life expectancy → **Difficult**

% Drop per Hour - Per Device

Device A - % battery drop per hour

80 - 78 = 3%
72 - 67 = 5%
25 - 23 = 2%
23 - 74 = -51% | **DISCARD**
50 - 48 = 2%
10 - 7 = 3%

3% average battery drop per hour

Discard readings where charger & power event took place

$$\frac{100}{Average\ \%\ drop\ per\ hour} = Projected\ Hours\ of\ Battery\ Life$$

% Drop per Hour - Entire Fleet

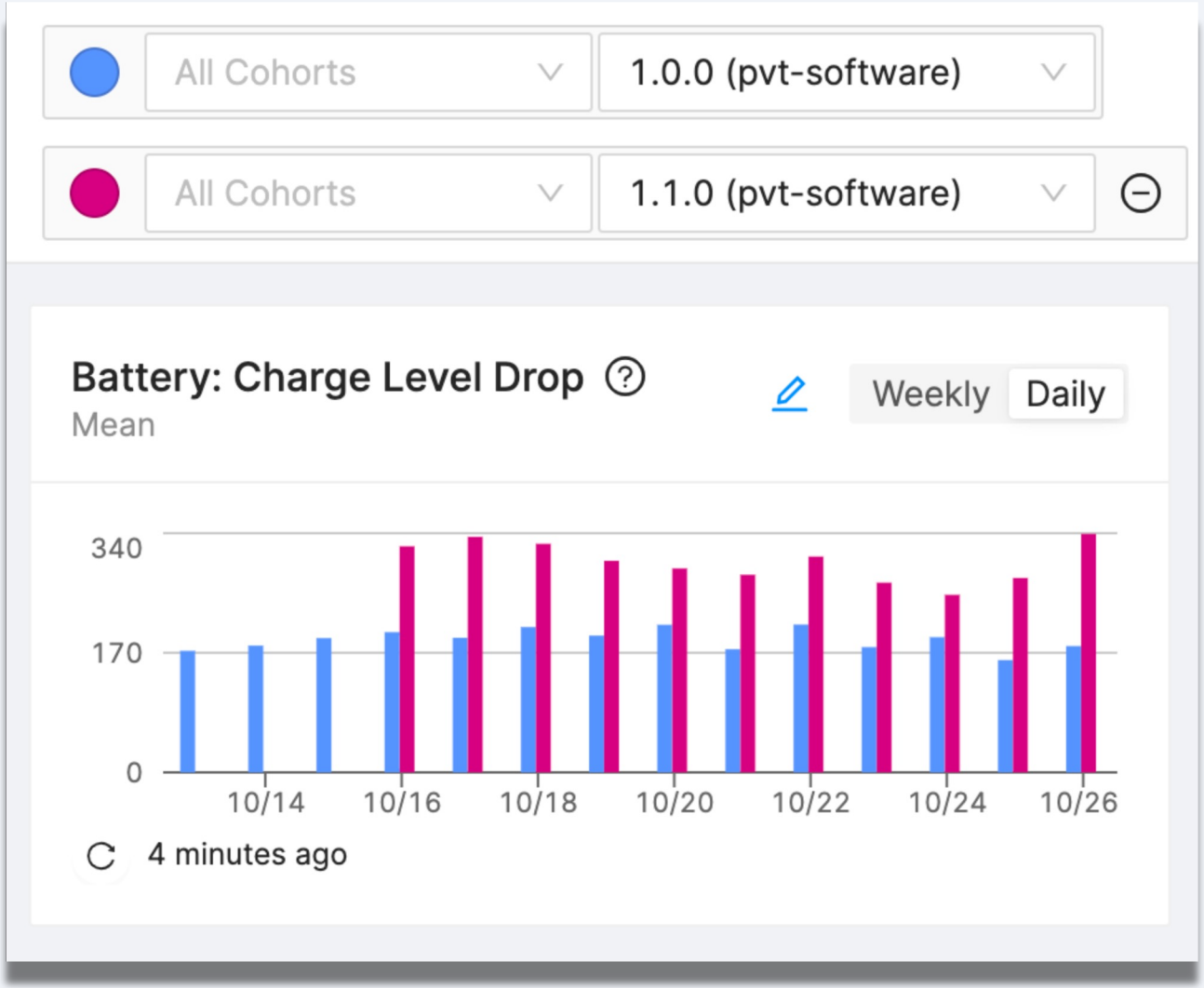
Device A	Device B	Device C	Device D
3%	2%	1%	2%
2%	3%	2%	2%
1%	1%	1%	1%
4%	3%	5%	2%
...	...	...	...

Average

$$\frac{100}{Average\ \%\ drop\ per\ hour} = Projected\ Hours\ of\ Battery\ Life$$

Easily works with an **unlimited number of devices**

Spotting Battery Regressions



Battery life expectancy was 2x shorter in 1.1.0!

Capturing Battery Drop per Hour

```
void memfault_metrics_heartbeat_collect_data(void) {
    ...
    static int32_t s_prev_battery_pct;

    const int32_t current_battery_pct = battery_get_pct();
    const int32_t battery_delta =
        current_battery_pct - s_prev_battery_pct;

    memfault_metrics_heartbeat_set_unsigned(MEMFAULT_METRICS_KEY_DEFINE(BatteryPctDrop),
                                            battery_delta);

    memfault_metrics_heartbeat_set_unsigned(MEMFAULT_METRICS_KEY_DEFINE(BatteryPct),
                                           current_battery_pct);
}
```

https://docs.memfault.com/docs/best_practices/metrics-for-battery-life/

Smart Sink

Devices

Create Devices using CSV

Device Serial | Cohort | Nickname | Software Version | Hardware Version | Last Seen | Staged | Custom Attribute

Historical Data between 2022-10-26 → 2022-11-02 PDT

Timeseries: battery_perc_delta > 20

Timeseries | Rebooted due to...

Historical Data

Change Cohort | Save as Device Set (36) | Reset filters

36 Devices	Cohort	Nickname	Software Version	Hardware Version	Last Seen
MFLTXX0001	Internal		1.0.1	xx_evt	18 hours ago
MFLTXX0002	Production		1.0.1	xx_evt	18 hours ago
MFLTXX0004	Production		1.0.1	xx_evt	18 hours ago
MFLTXX0005	Production		1.0.1	xx_evt	18 hours ago

Start 2022-10-26 00:12:00.000

End 2022-11-01 22:12:00.000

Thu 27 | Fri 28 | Sat 29 | Oct 30 | Mon 31 | November

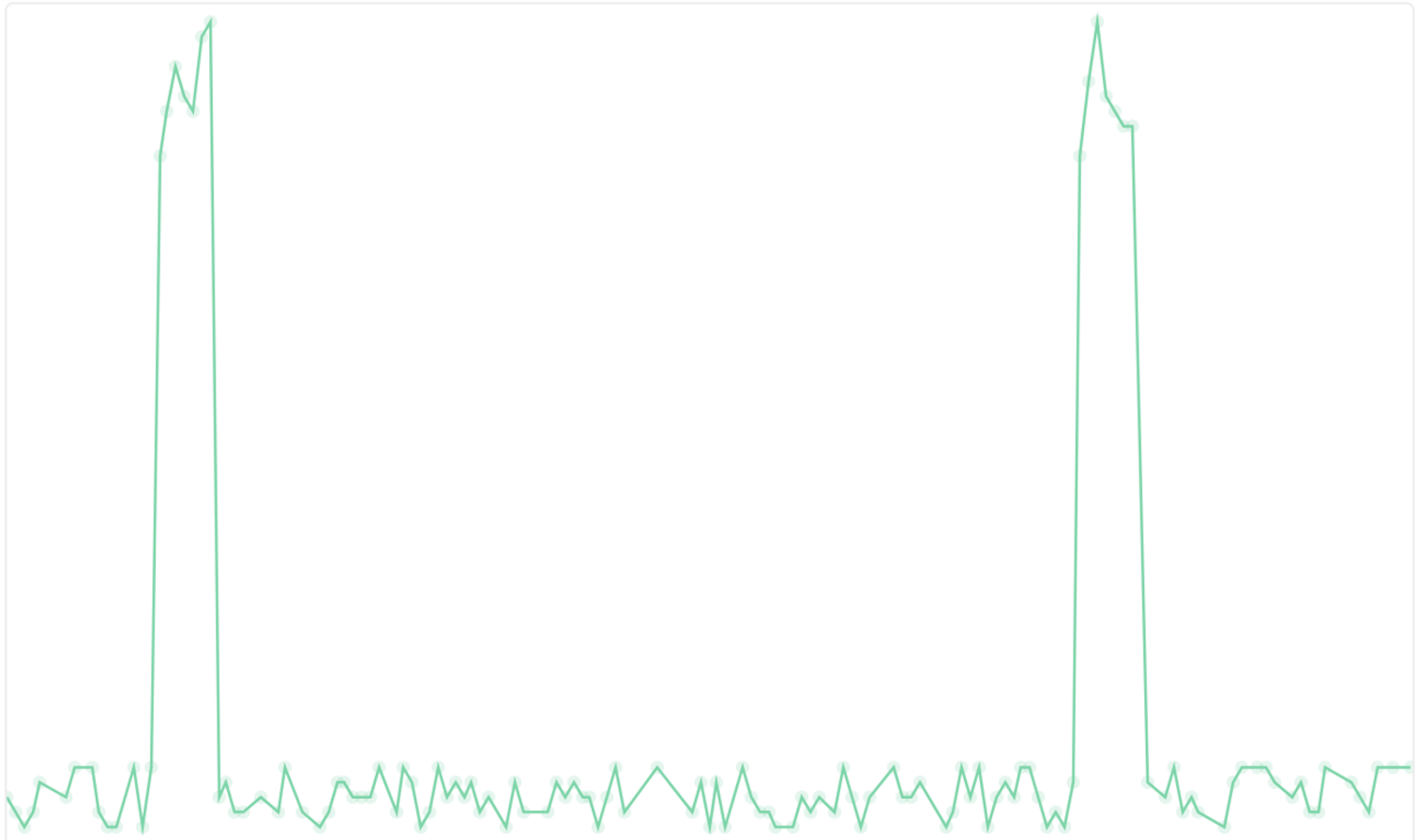
Coredumps

Log Files

> Reboots

Traces

battery_perc_delta



Start	End
2022-10-26 00:12:00.000	2022-10-26 23:09:10.979 2022-11-01 22:12:00.000

Thu 27 Fri 28 Sat 29 Oct 30 Mon 31 November

Coredumps

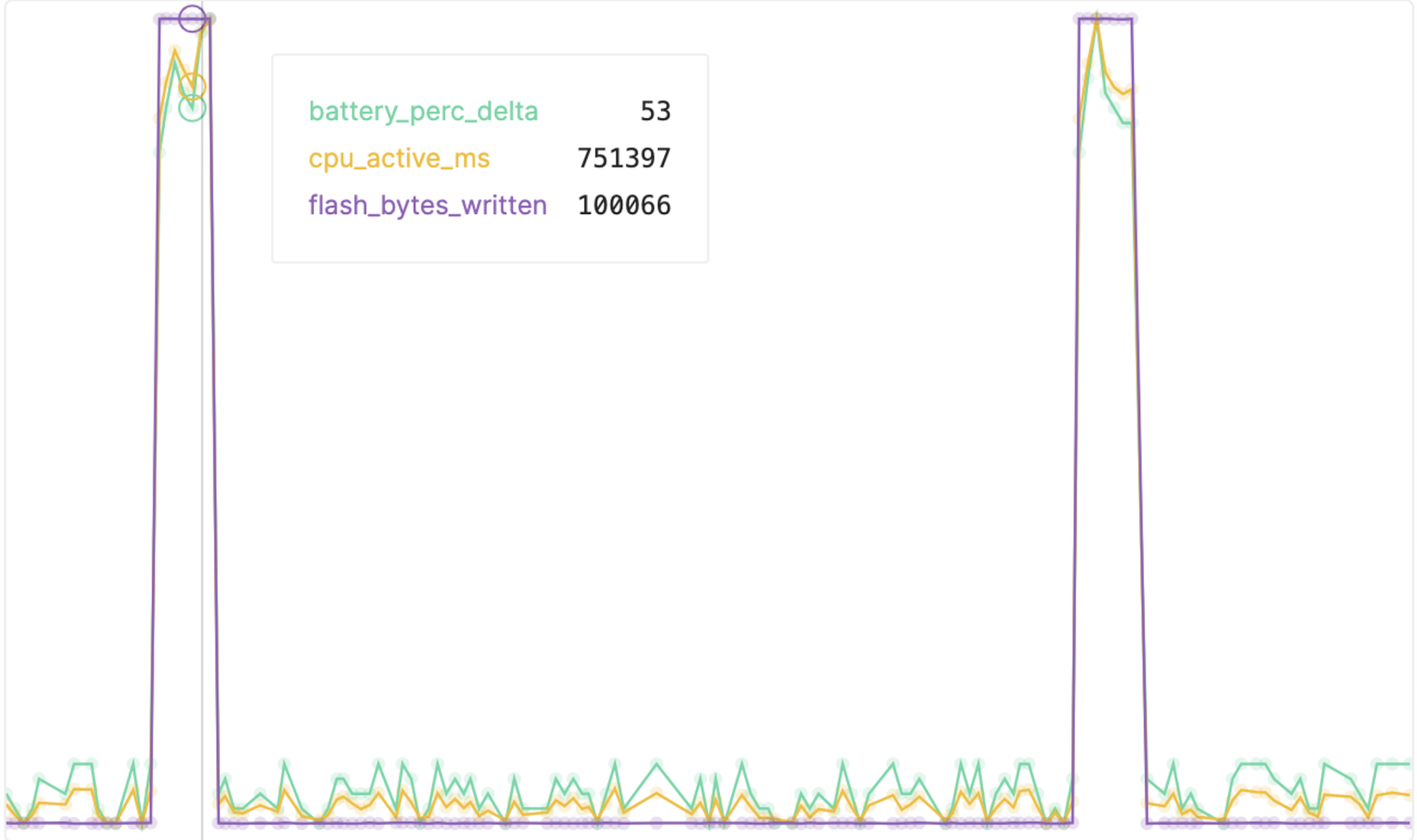
Log Files

> Reboots

User Reset

Traces

3 inputs

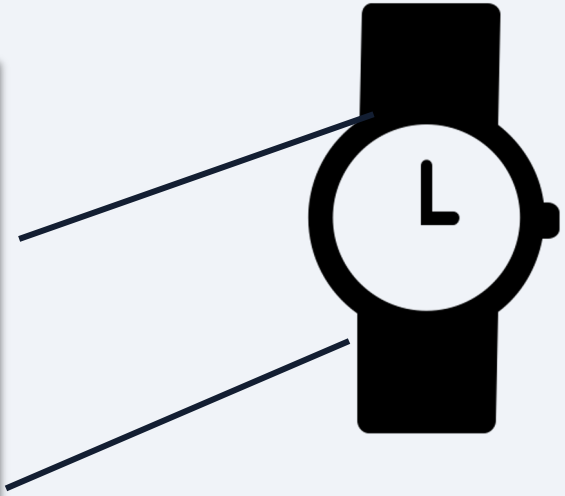


Agenda

1. What & Why of Metrics
2. Battery Life Metrics
3. Power Consumption Metrics
4. Connectivity Metrics
5. Best Practices

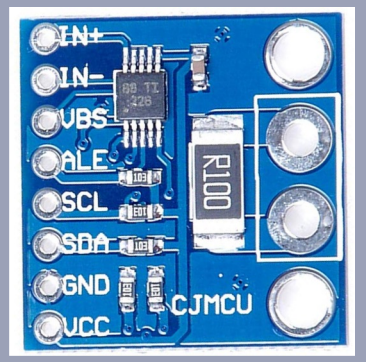
Power Consumption Metrics

Measuring Power Consumption - Locally



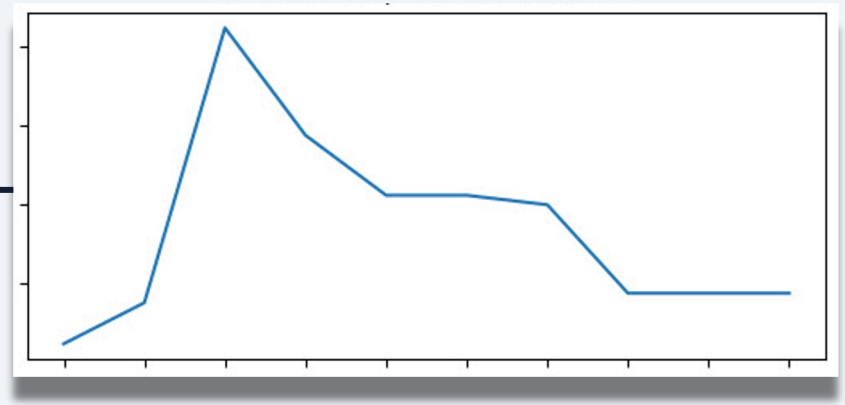
Using a multimeter - locally

Measuring Power Consumption - Locally



TI
INA-226

USB

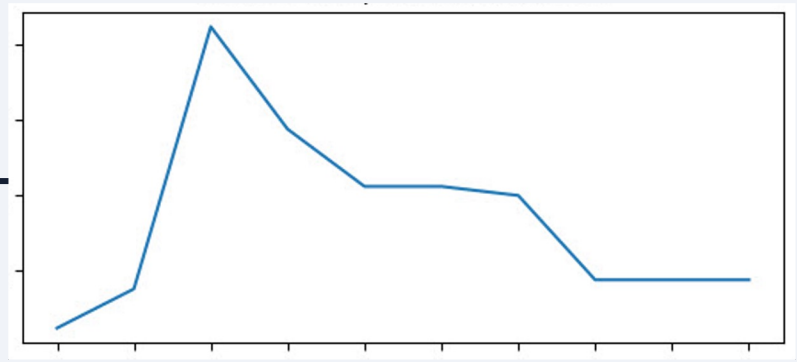


Using a built-in power monitor - locally

Measuring Power Consumption

Integrated Fuel Gauge

USB



Counting coulombs using a fuel gauge

Power profiles for peripheral usage

- Run with peripheral powered off
- Measure power of peripheral **disabled**
- Instrument & exercise peripheral
- Measure power of **busy peripheral**
- Take the delta of power consumption

Estimate power draw over time or by usage

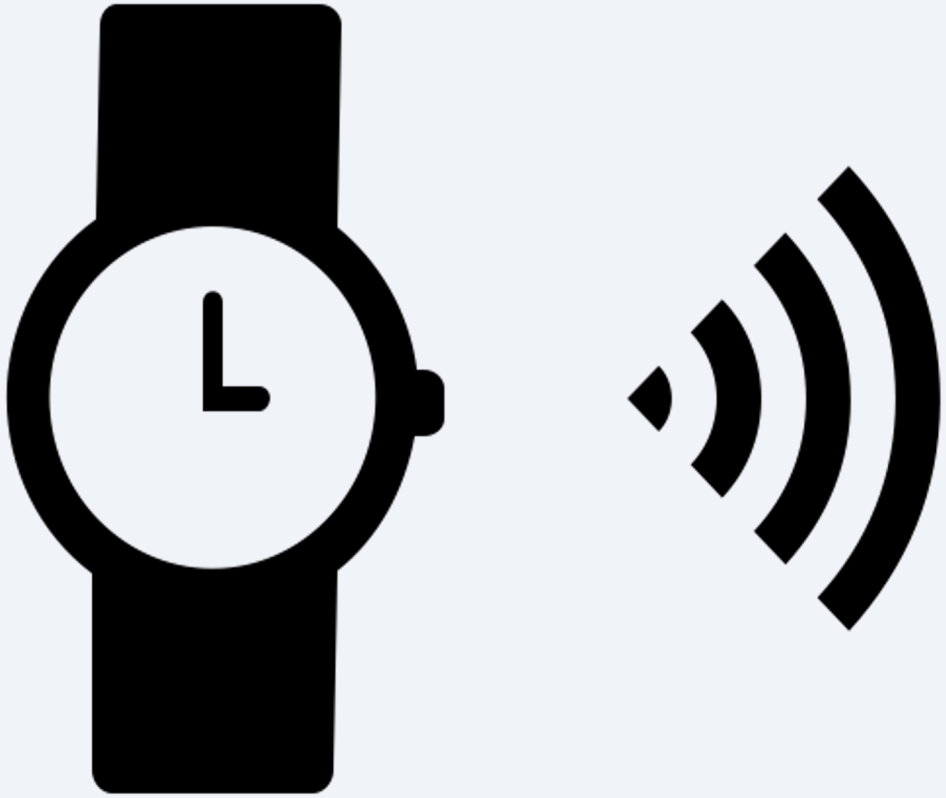
µAh per external flash sector erase

µAh per BLE minute scanning

µAh per second LCD backlight on

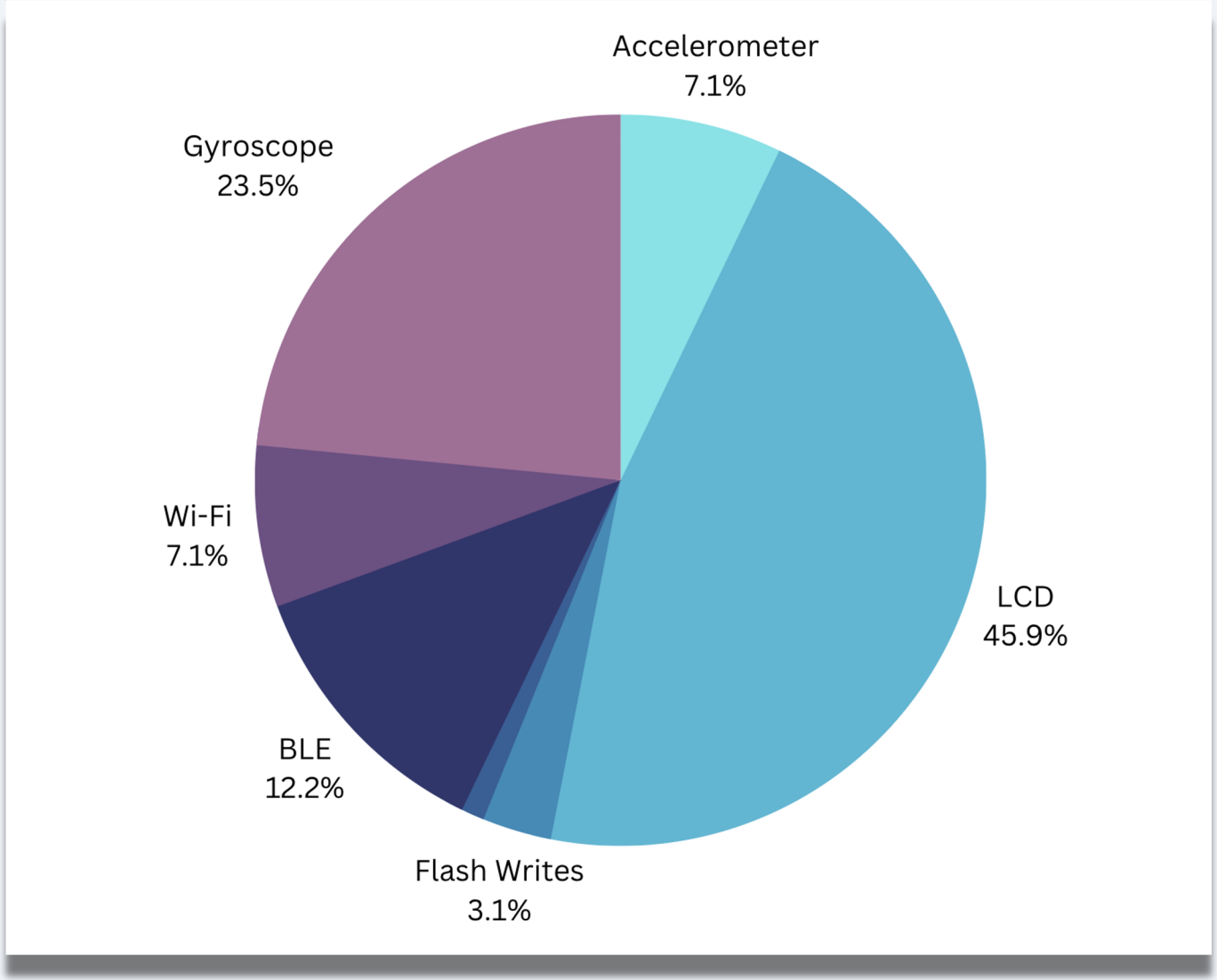
µAh per second vibe motor

Measuring Power Consumption - *Remotely*

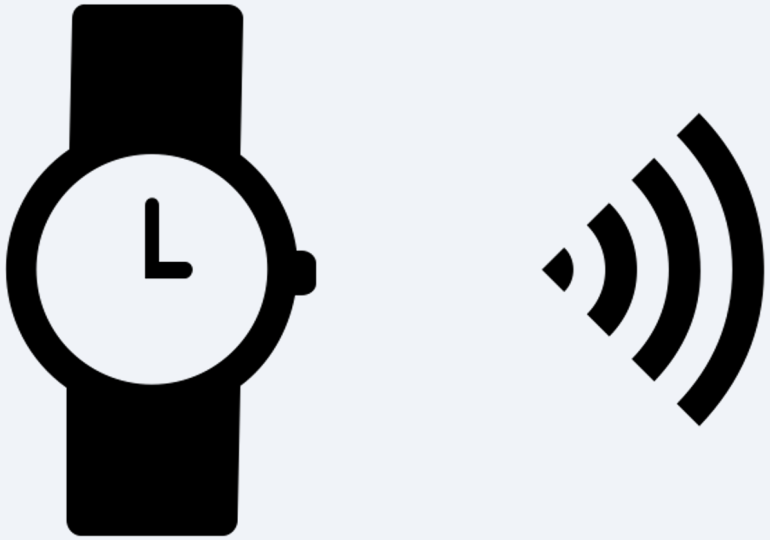


Measuring Power Consumption - *Remotely*

We want a distribution of where power is being spent



Measuring Power Consumption - *Remotely*



```
Example App Booting
Metric ID: 1 -- Value: 15000  # elapsed ticks
Metric ID: 2 -- Value: 8531   # Main Task ticks
Metric ID: 3 -- Value: 8223   # Timer Task ticks
Metric ID: 4 -- Value: 14     # Timer Task count
Metric ID: 5 -- Value: 3593   # "sensor on" ticks
Metric ID: 6 -- Value: 4696   # heap-free low watermark
```

Metrics of course

Power profiles for peripheral usage

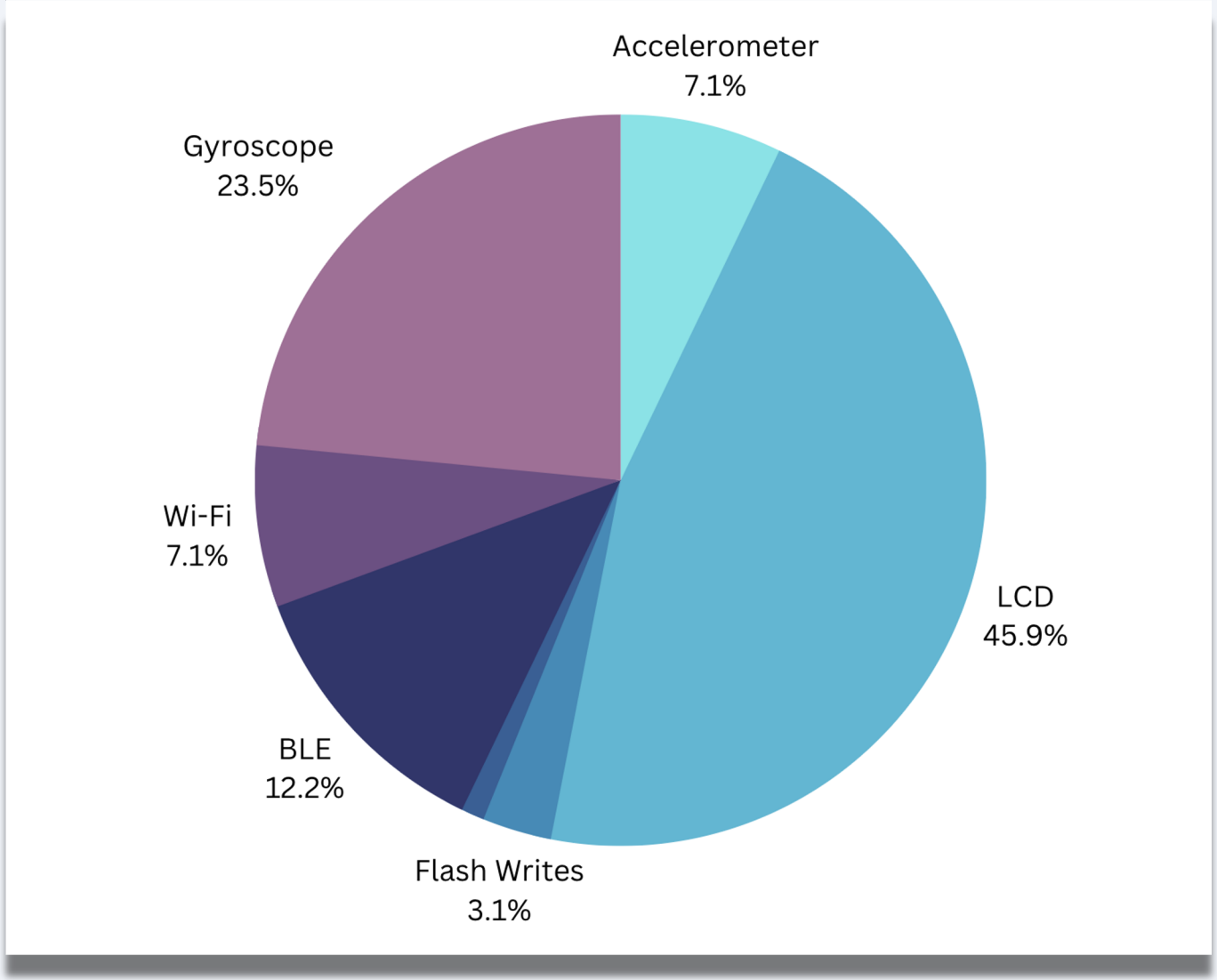
Metrics for peripheral usage

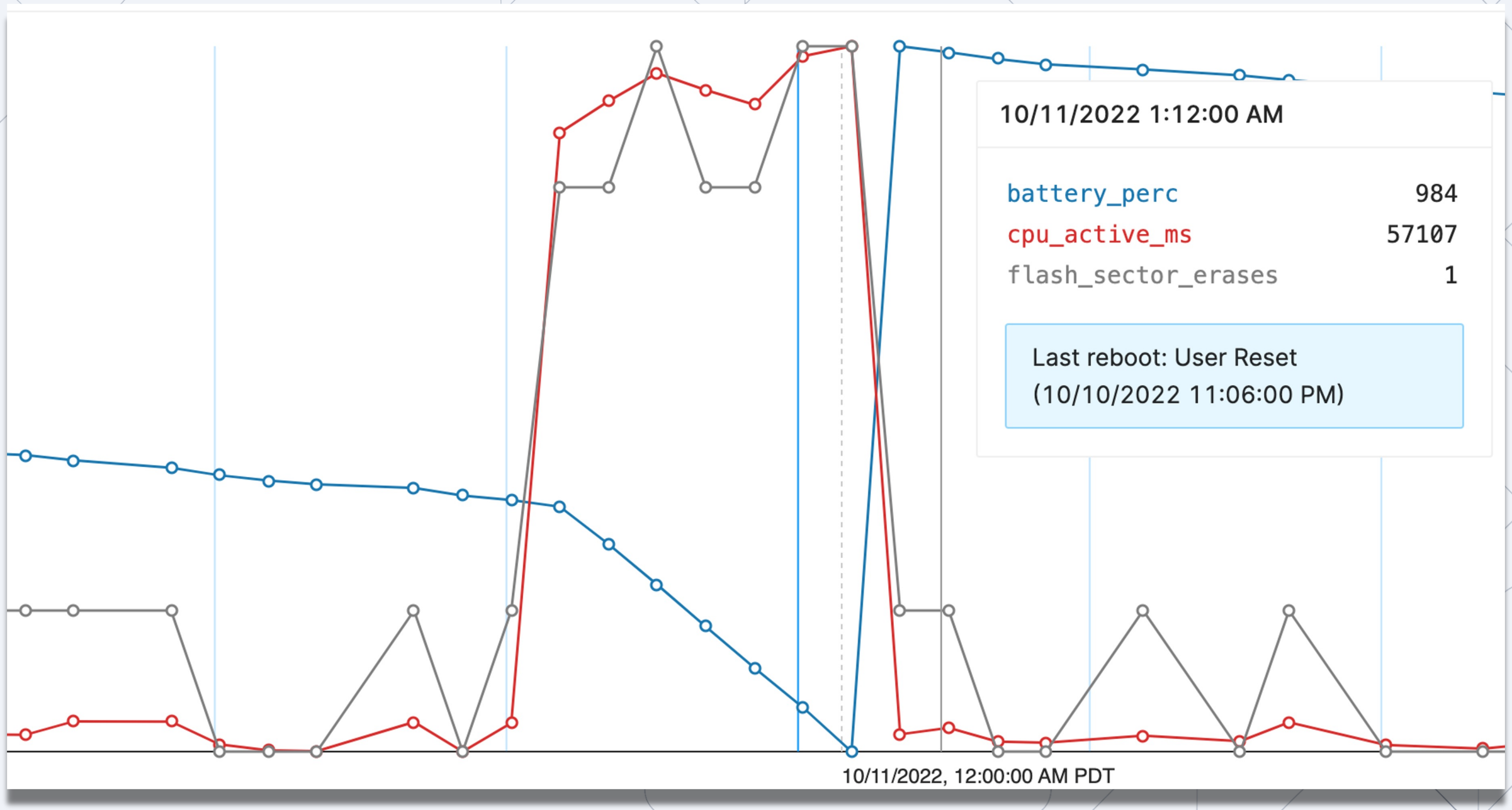
+

Power usage per “metric unit”

=

Crudest method what consumes the power of a battery




10/11/2022 1:12:00 AM
battery_perc 984
cpu_active_ms 57107
flash_sector_erases 1
Last reboot: User Reset
(10/10/2022 11:06:00 PM)
10/11/2022, 12:00:00 AM PDT

Spotting regressions in power consumption

- In production devices…it isn't easy
- First line of defense is battery tracking mentioned before
- Next, would have metrics for the common power hungry systems
- Determine "steady state" of these metrics across all production units
- Raise a flag if they change dramatically

Average time LCD Backlight On per Hour

1.2.1	121s
2.0.0	243s

Average time CPU Sleep Mode per Hour

1.2.1	3457s
2.0.0	2592s

Agenda

1. What & Why of Metrics
2. Battery Life Metrics
3. Power Consumption Metrics
4. Connectivity Metrics
5. Best Practices

Connectivity Metrics

Most devices have *some* connectivity

Most devices have *some* connectivity

- Min, Max, & Average Signal Strength (RSSI)
- Radio Channel
- Total number of connects and disconnects
- Total number of connect failures
- Total number of bytes Received & Transmitted (RX/TX)

```
void wifi_connected_callback(void) {

    // Start timer when Wi-Fi connects
    memfault_metrics_heartbeat_timer_start(...);

    // Count connection
    memfault_metrics_heartbeat_add(...);
}

void wifi_disconnected_callback(void) {

    // Stop timer when Wi-Fi connects
    memfault_metrics_heartbeat_timer_stop(...);

    // Count disconnection
    memfault_metrics_heartbeat_add(...);
}
```

Most Common Support Requests (BLE)

"I receive phone calls but not text messages"

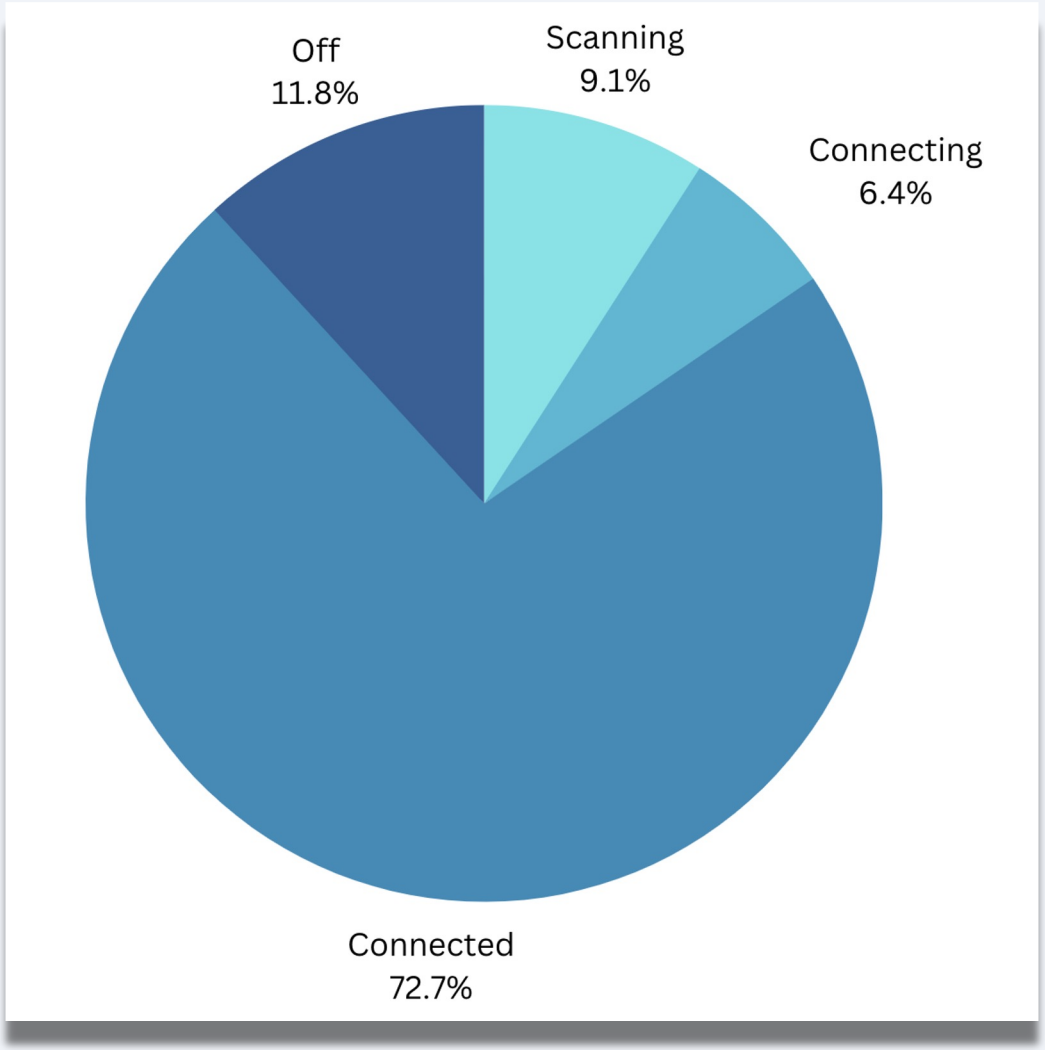
"It doesn't work"

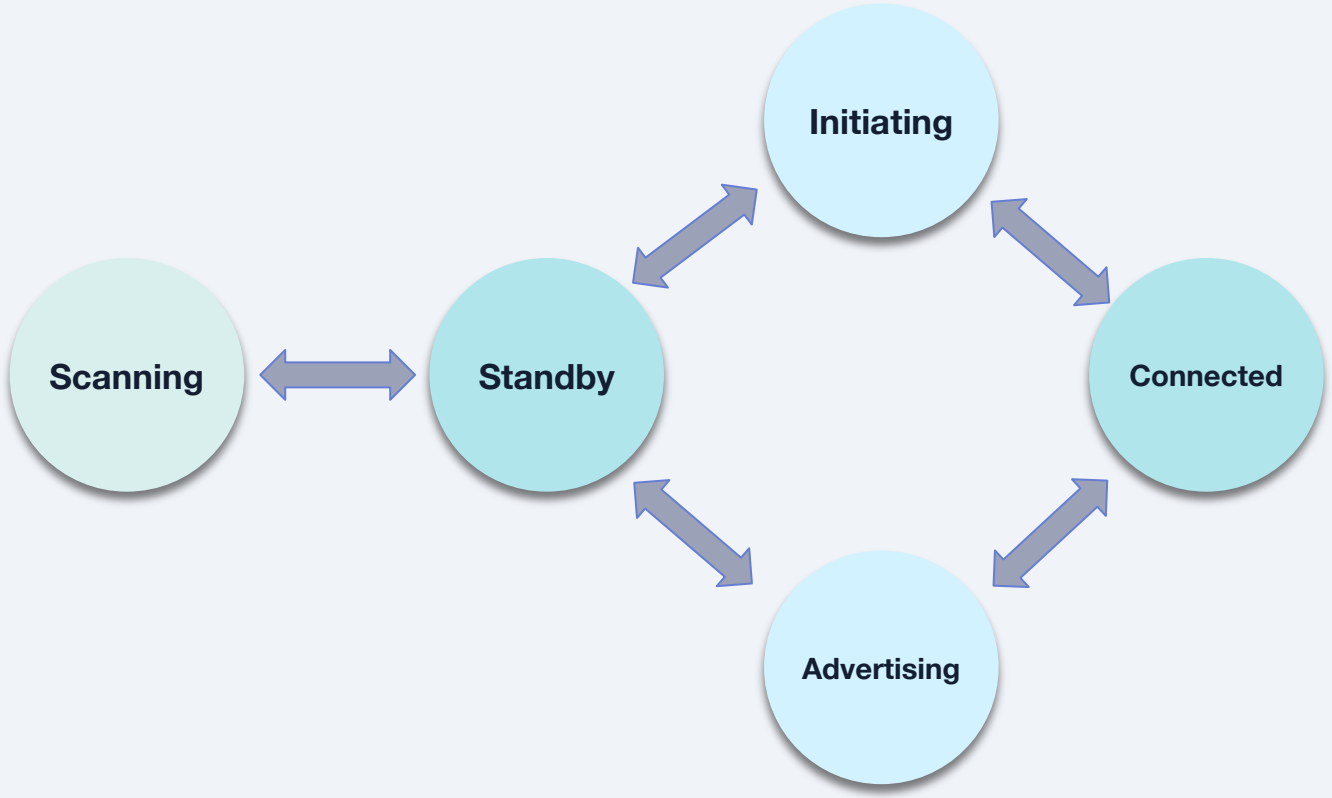
"It doesn't stay connected"

"Poor battery life when connected to *<random Android phone>*"

"My Pebble's Bluetooth doesn't connect to my 2012 Mercedes"

Distribution of Connectivity States





Aggregating Connectivity Metrics

Device A	Device B	Device C	Device D
90%	85%	83%	93%
87%	79%	78%	91%
60%	99%	77%	95%
84%	40%	90%	98%
...	...	...	...

$$\frac{Total\ time\ connected - all\ heartbeats}{Total\ possible\ time\ connected - all\ heartbeats} \times 100 = \%\ Connected$$

Easily works with an **unlimited number of devices**

Agenda

1. What & Why of Metrics
2. Battery Life Metrics
3. Power Consumption Metrics
4. Connectivity Metrics
5. Best Practices

Best Practices

Poll #2

If you had to choose…

which type of issues does your firmware suffer from most?

A. Battery & Power

B. Connectivity

C. Performance

D. Stability & Crashes

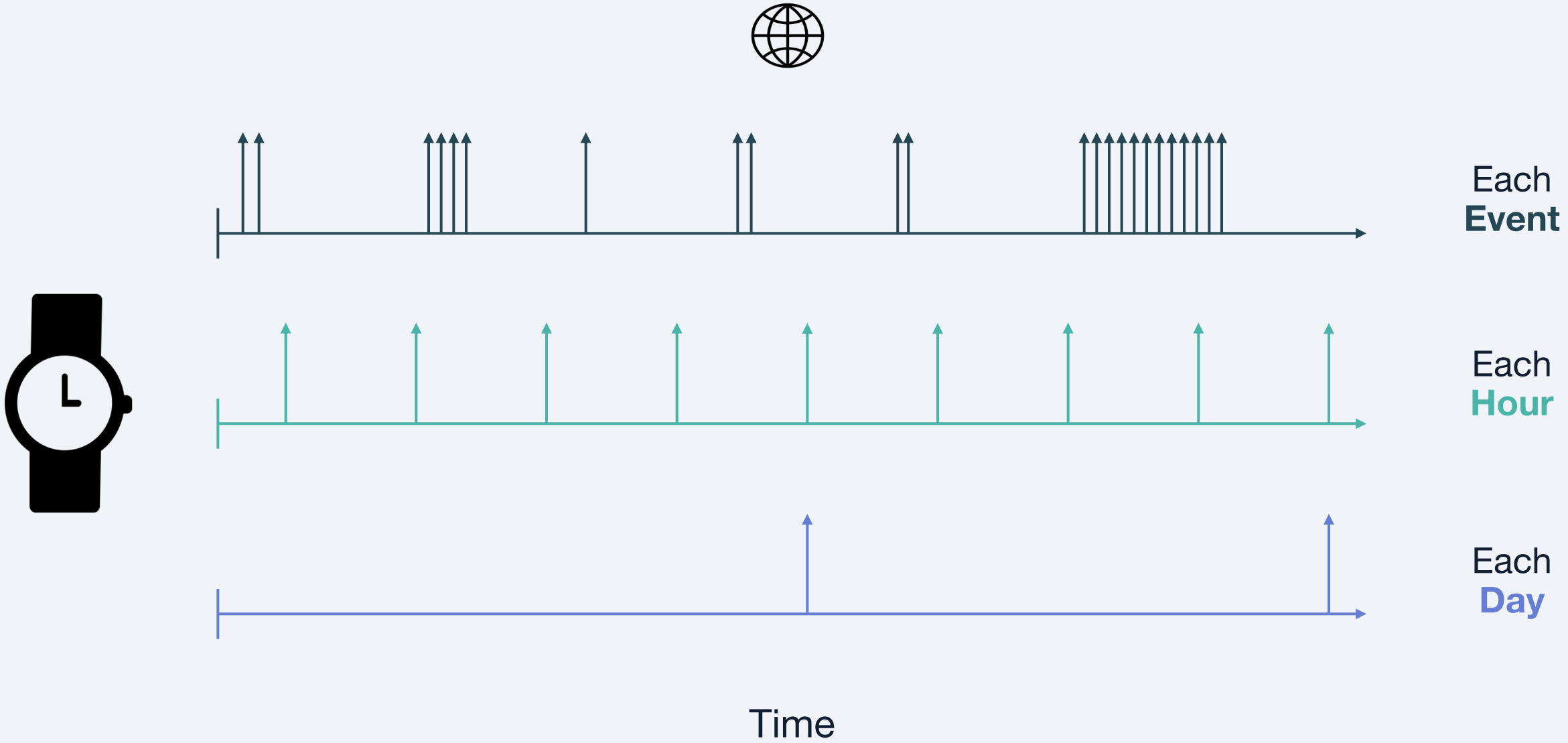
Does Collecting Metrics Impact Battery Life?

Yes and No

- Metrics are just incremental counters tracked in RAM.
- No difference in power consumption compared to other items (heaps, queues, RTOS operations)

- Don’t log KB and MB of ASCII
- Flush metrics to flash each heartbeat interval
- Send metrics when power allows for it

Does Collecting Metrics Impact Battery Life?



Does Collecting Metrics Impact Battery Life?

Ultimately, without metrics, battery life suffers will suffer with products that are anything but trivial.

Any one of us can and will commit issues relating to drivers, peripherals, sleep timeouts, etc.

Store & Send Metrics Efficiently

```
{
  "schema_version": 1,
  "device_serial": "D12345678",
  "software_type": "nrf52",
  "software_version": "1.0.0",
  "hardware_version": "evt",
  "event_info": {
    "BleAppConnectTimeMs": 1000,
    "BleAvgConnIntervalMs": 200,
    "BleAvgRssi": -54,
    "BleBytesTransferred": 20000,
    "BleConnParamChangeReq": 10,
    "BleCrcErrors": 4,
    "BleDisconnectedTimeMs": 600000,
    "BleMicErrors": 0,
    "BleNumDisconnects": 3,
    "BleStackHwm": 3000,
    "BleTxBufferFullErrors": 0,
    “IntervalMs": 3600000,
    "RebootCount": 0
  }
}
```

Encoding	Example Payload Encode Size (bytes)
JSON	471
MsgPack	399
Protobuf	72
packed c struct	69
Memfault	68

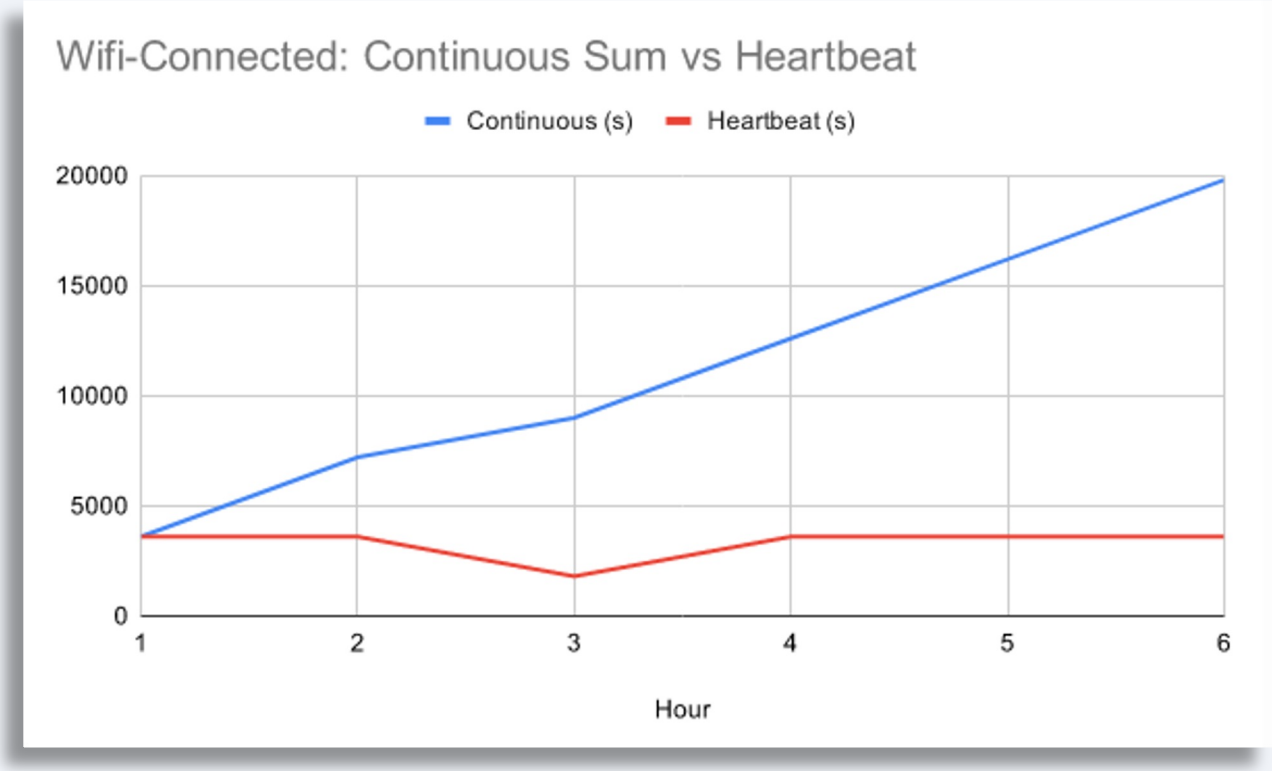
- Strings & metric keys are embedded within symbol file
- CBOR encoding
- Compress integers

- Less RAM
- Less CPU encoding
- Less bandwidth
- Less time sending data
- **Less power consumption**

Memfault Documentation: https://docs.memfault.com/docs/mcu/event-serialization-overview/

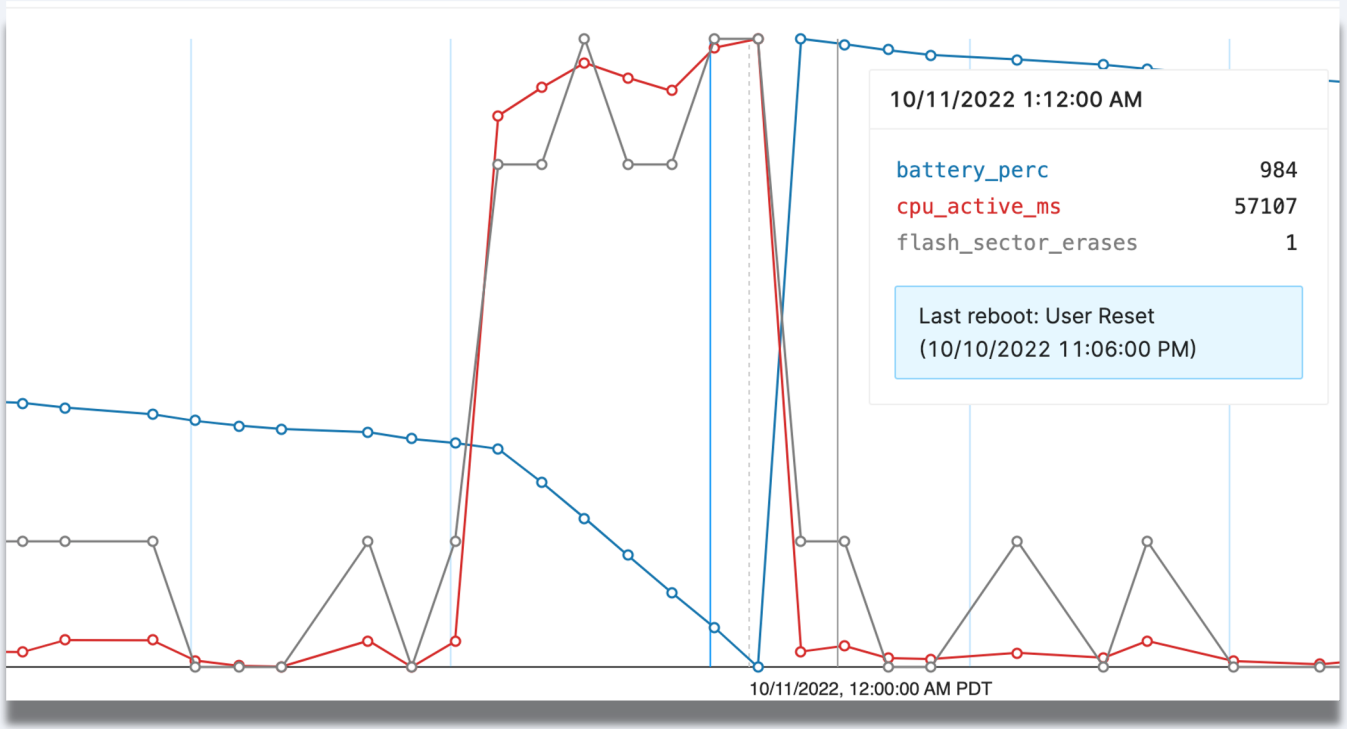
Reset Metrics each Heartbeat

- Easily spot issues and trends per device and across fleets
- Resilient against resets and data loss
- Simpler math for everyone and especially tools & databases



Metrics for a Single Device

- Should be plotted on a timeline
- Can show correlation between various metrics
- Essential for customer support
- Great for debugging “It doesn’t work”
- Surfaces subtle regressions and odd behavior



View Stats About Fleet of Devices

View metrics about production devices

Mean % battery drop per hour

1.2%

Average projected battery life

3.47 days

Average time connected to Wi-Fi

83%

Average frames per second

19 fps

View Stats About Fleet of Devices

We can compare metrics between firmware versions

Mean % battery drop per hour

1.2.1 1.2%

2.0.0 1.1% ↑

Average projected battery life

1.2.1 3.47 d

2.0.0 3.79 d ↑

Average time connected to Wi-Fi

1.2.1 83%

2.0.0 92% ↑

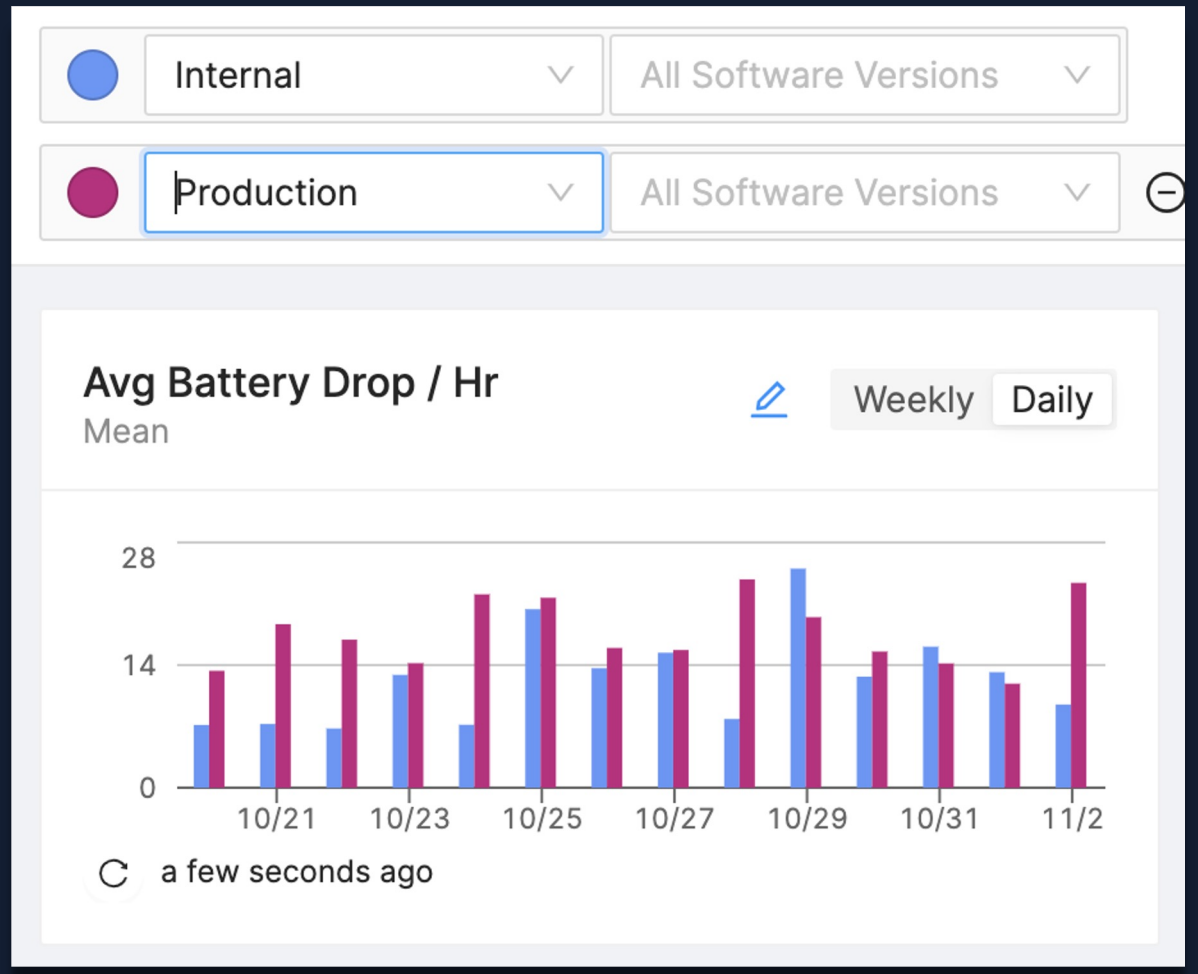
Average frames per second

1.2.1 19 fps

2.0.0 13fps ↓

Other Best Practices

- Persist immediately and send heartbeats when possible.
- Compare metrics primarily between software versions & other dimensions. Not over time.
- **Quickly iterate on metrics**



Agenda

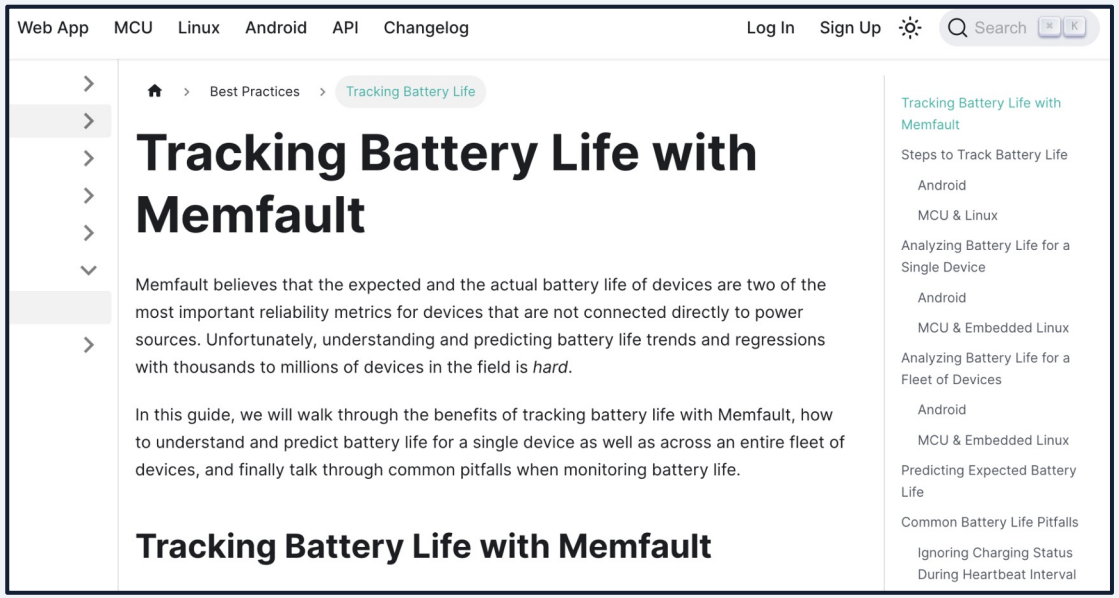
1. What & Why of Metrics
2. Battery Life Metrics
3. Power Consumption Metrics
4. Connectivity Metrics
5. Best Practices

Thank You!

Memfault

December 8th - Building Out a Firmware Team

Interrupt Panel with Phillip Johnson from Embedded Artistry



https://docs.memfault.com/docs/best_practices/metrics-for-battery-life/

memfault.com

twitter.com/memfault

linkedin.com/company/memfault/

Q&A

Would appreciate filling out the survey at the end.

It will appear in browser when the webinar is over and we will have it in the follow-up email.